IT Middleware Services for an "Exploration Web"

Norm Lamarra
Jet Propulsion Laboratory/Caltech
Pasadena, CA 91109
Norman.Lamarra@jpl.nasa.gov 818-393-1561

Abstract—Distributed applications in business, engineering, and management are evolving rapidly via the concept of "web services" delivered over the Internet or a business intranet. A vision is emerging of a future "connected economy", in which businesses are able to maintain and leverage success by responding rapidly to changes in demand for customer services. Space applications, however, face a different set of constraints, some of them much harder to manage. However, we believe that the concept of shared services is equally valuable in space, and we believe similar adaptability is feasible for space applications. The key to this is building middleware layers on top of standards-based communication, thus providing adaptability (in the middleware functionality) without violating communication standards. The result is that participating spacecraft can more easily produce or consume diverse kinds of information and services, such as navigation, weather, terrain, and remote computational capability. As more infrastructure services are deployed and more craft co-operate, the environment becomes progressively richer, yet participation can be made extremely simple (e.g., for low-cost "sensor web" micro-units). Our approach improves the transparency of information, allowing missions to leverage each others' resources/capabilities and hence participate in an evolving "exploration web" with significantly more capability for efficient, automated, and even autonomous remote exploration. In such an "exploration web", rovers, landers, and orbiters would be able to collaboratively utilize all assets more effectively and robustly, potentially providing much greater science return with lower operations cost and risk. Further in the future, we envision being able to build or modify applications robustly "on the fly" when the software components and data sources may be in different locations, languages, formats, etc. This extends the usefulness of space assets by enabling them to benefit from the evolving technology cycle.

TABLE OF CONTENTS

1. INTRODUCTION

JPL is developing an Interplanetary Network (IPN) in order to extend the reach and benefit of capabilities that we have come to use routinely in the terrestrial Internet [1,2,3]. The vision is grand but clear – spacecraft as "nodes on the Internet", web-accessible as distributed resources, albeit ones with certain differences from those on Earth or in near space. This paper starts from the assumption that a) this is a good future goal; and b) many of the required technologies are already in use today on the ground. We thus address an approach toward leveraging those technologies for use in space, namely space middleware services. In this context, middleware is viewed as software involved in connecting separate application pieces (components), but excluding that involved in the communication connection itself. Indeed, a primary goal of such middleware is to provide distributed capability that hides the vagaries of that communication. The middleware can thus be viewed as residing between application components (layered "above") and communication components (layered "below"), as shown in Figure 1.

The goal of space middleware is to simplify cooperation between spacecraft over many missions. It is less obvious why this would be of benefit for an individual isolated mission, but it becomes more obvious as more craft are involved. For example, at this date there are 5 missions about to launch for Mars over the next few months, and others are already operating in Mars orbit. Apart from this kind of opportunity, there are also future plans to fly cooperative spacecraft, even some in tight formation. It may thus be a good time to consider the potential benefits of space middleware in the IPN context.

Assuming there are other spacecraft reachable in the IPN, space middleware can assist each to extend its own capability by leveraging the resources of others (e.g., remote storage, computation, or connectivity). Resource sharing is not a new concept, but has not been well developed in this context. Middleware-enabled sharing implies simplified access (e.g., via standard API's) built upon persistent capability (services) designed to deal with fundamental limitations imposed by the constraints of space operations. Such constraints include: episodic connectivity; extreme power limitation; low-bandwidth and long-latency links; need for unattended operation.

A simple example of such a service accessed through a standard interface is the CCSDS File Delivery Protocol (CFDP), which enables efficient file transfer over such space links, and conceptually replaces the standard Internet File Transfer Protocol (FTP). Viewed from the perspective of enabling middleware, CFDP hides the constraints of the space link by appropriate use of buffering, accounting, and retransmission, while guaranteeing complete and ordered delivery of data in file units (as opposed to frames or a bit stream). More generally, it can be viewed as an internet "service" designed to succeed when standard Internet protocols are inadequate because one or more of their design assumptions (like continuous connectivity) are violated.

Some potential benefits of a robust space-based middleware approach are:

a) simplified space applications;
b) increased automation;
c) reduced cost;
d) increased reliability and robustness;
e) ability to evolve capability with little or no modification of the applications.

A major issue is to define what persistent capabilities would provide most benefit to most space applications at the lowest cost, and then to develop and deploy them in the optimal order. Rather than attempting to reach a broad consensus on this *before* starting down this path, our approach is more pragmatic – namely, to quickly demonstrate some initial simulated capability to otbtain feedback and perhaps to gain support for extending it progressively. We have so far addressed two example space applications. The first application addresses the

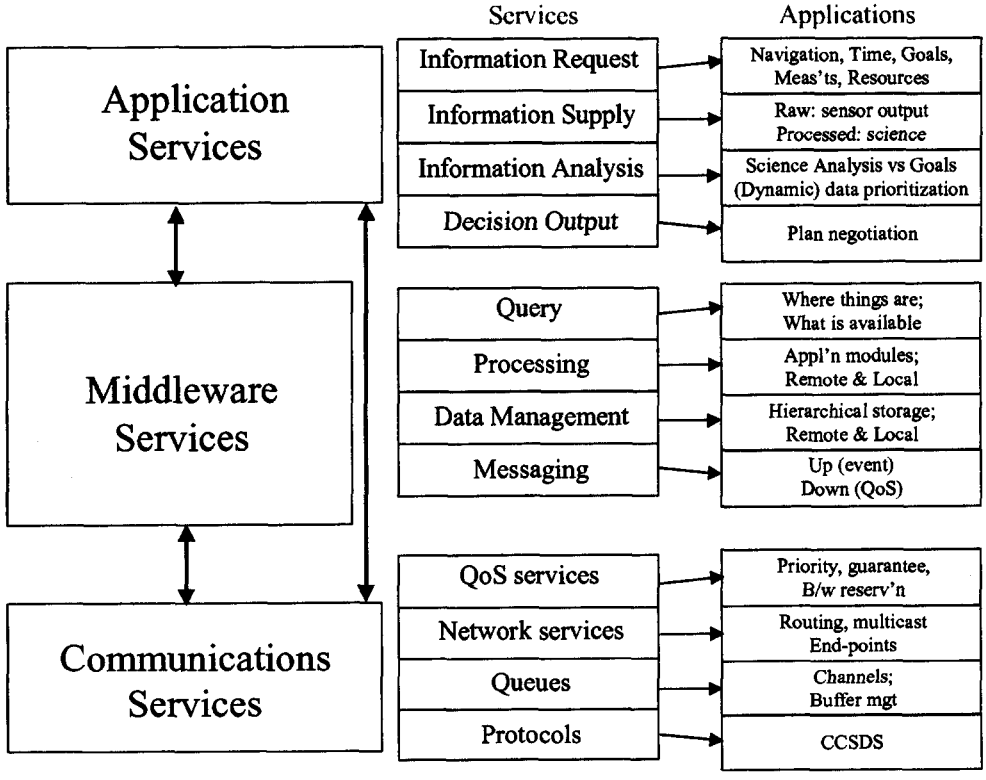

Figure 1: Layered View of Space Middleware

common difficulty of intermittent or pass-oriented communications, i.e., the desire to utilize more connection paths (even unknown ones) in a seamless way. The second addresses the fact that sensor bandwidth can significantly exceed downlink bandwidth. The reader can judge whether the demonstrated approach achieves the general benefits desired, particularly in simplifying the applications. More importantly, it needs to be decided when the service approach can provide sufficient return on investment. Some lessons learned from the evolution of the Internet may be directly relevant:

a) The Internet did not evolve primarily by prior coordinated design but by increasing usage and incremental (experimental) extension – successes evolved new standards;
b) The evolution of today's wireless web is dealing with many issues similar to those mentioned above for space (e.g., disconnectedness caused by mobility, extreme power limitation as in handheld devices, etc.);
c) The hot topic of "web services" promotes a view of distributed applications that supports the above middleware approach;
d) Costs of participation reduce dramatically with widespread acceptance via standardization.

2. ENABLING TECHNOLOGY

We have looked at two key enabling technologies that support middleware for space: robust asynchronous messaging; and a shared object model. We assume that communications services are available and perhaps standardized (e.g., CCSDS protocols).

Robust messaging addresses many of the limitations imposed by the space domain, such as disconnectedness, long-latency links, etc. Moreover, message-oriented middleware (MOM) has a successful history in the evolution of publish/subscribe internet applications, as distinct from those built using tightly-coupled remote invocation. At a high level, this difference can be compared to that between sending email and having a phone conversation. The former allows the sender and receiver each to continue performing work independently of the (background) email transfer, while the latter requires both parties to be interacting simultaneously in real time (with some acknowledgment). The ubiquity and volume of today's email traffic attests to its utility as a productivity aid in the work environment. Moreover, in the space domain it is often impossible to provide real-time interactivity. Messaging can also assist automation, since tasks can be linked to the content of such messages. As a simple example, action might be triggered only when the value of a certain field received in a regular message reaches a certain range.

This leads to discussion of the second enabling technology: a shared object model can enable such messages to have significantly more meaning. An object of a particular type encapsulates specific properties and coded functions; if an object of this type is created by one application then an similar object can be created and manipulated on a remote platform without transferring the entire object (which may be quite large). Separate application components (e.g., on different platforms) can thus interact efficiently because there are standardized rules for each to deal with the referenced objects. An example of this in the space domain is that a model of a sensor can be created on the ground and can contain functions like "take current reading" or "set calibration value". While the actual sensor may be on a remote spacecraft, ground software can interact with the local model of the sensor in exactly the same way as the spacecraft can, provided the ground and flight software share the object model. More importantly, these objects can be coupled by messages containing only the object reference and the property values, thus providing an efficient way to structure communications. A more futuristic example application could embed the object in a virtual world accessible to ground participants without requiring a real-time connection. This addresses (and extends) the notion of interacting with spacecraft as "live" nodes on the internet, but also shows a way to significantly simplify interactions between spacecraft.

3. EXAMPLE APPLICATIONS

We have explored two prototype applications using these key technologies to explore the ability of messaging middleware either to simplify the application or to support new capability. These prototypes were not chosen to be currently realistic, but to show some of the potential utility of the middleware approach in constructing possible future scenarios with greater capability or flexibility. With such a controversial approach (space middleware), an abstract example can appear too general to be useful, while each specific example may appear not to be addressing needs relevant to a particular reader. They should thus be viewed as a way for us to explore potential benefits and pitfalls of shared space middleware.

The first prototype application addresses a simulated scenario involving communication between a rover and the ground. Normally, this communication is scheduled for a single pass per sol, during which telemetry is downlinked and commands are uplinked. In that case, only simple point-to-point scheduled communication is necessary, but conversely this limits the rover to a 1-sol

cycle of reporting and plan updates. Our middleware prototype added consideration of another craft (orbiter) as an ad-hoc relay, thus allowing a more rapid cycle. However, in order to investigate the concept of a flexible relay, we postulated that some adaptivity would be required of the joint activities of the rover and orbiter. This adaptivity could take the form of a request by the rover for relay capability to Earth (from any possible responder), and a negotiation between the responder for changes to both their plans to accommodate the request.

JPL has had significant experience developing a robust object-based messaging middleware for military application. Since many of the constraints of this application are similar to those described for space, we attempted to apply this middleware (SharedNet) to explore the above scenario. We thus created a new activity type in the (shared) object model and used the messaging middleware to transfer attributes (such as task type, start and end time) between existing planner applications assumed to be running on both rover and orbiter. The communications model between SharedNet participants is described by clients publishing messages to a server and registering subscriptions to particular object types. In the rover/orbiter example, the orbiter could host the server, and messages between the rover and orbiter would be transferred in prioritized order when connection between them was established (i.e., when the rover and orbiter plans allowed it). Subsequent plan negotiation results from each planner subscribing to plan objects from the other; these objects contain either plan update requests or replies. The demonstration showed that the publish/subscribe messaging middleware was easily adapted to this new (planning) application.

At first glance, such an approach might appear to be "overkill" for the type of scenario described, since a specific relay connection could be built into both rover and orbiter plans by agreement between the mission operations teams on the ground. However, the purpose of this investigate was to explore features that provide a framework for much greater potential capability, if the existence of appropriate middleware is considered. For example, it indicates that spacecraft communication can be automated (like cell-phone connectivity) to make better use of available communication opportunities and bandwidth. It also indicates how such connections can be efficiently utilized via a shared object model, and avoids the creation and implementation of a new application protocol for each such type of communication. Next, the use of a publish/subscribe paradigm allows participants to automatically obtain only the information they wish to receive when connection occurs, and show that many different kinds of information transfer can utilize the same connection (but different objects). For example, the messages could contain sensor readings, navigation information, images, etc., and could be destined for many kinds of users. Lastly, the subscriber may even be unknown to the publisher, for example an in-situ sensor publishes a regular measurement to a server which then makes them available to interested parties.

Having outlined some of the ways such a messaging infrastructure could be used, it can now be inferred that new capabilities can be conceived and implemented much more simply. For example, an automated application (e.g., agent) could collect information from a number of sensors (by subscription, without needing direct contact with any of them), and then perform a task based on derived information – e.g., to capture a seismological or weather event. The relatively simple remote automated messaging infrastructure thus opens the door to other event-driven science that might be impossible if round-trip connections to Earth were needed. From the ground viewpoint, downlink bandwidth might only be utilized when particular events are observed by sensors within the planetary network. This is becoming a sketch preview of the middleware-based "exploration web".

The second example application looked at the work of another research team at JPL (OASIS). In that application, images taken by a camera for navigation during long traverse are processed onboard to determine science value against criteria specified by the scientist. If "sufficiently interesting" results are obtained, then notification is sent to the ground at the next opportunity, perhaps resulting in the scientist requesting plan modification to interrupt the traverse in favor of a closer inspection of the "found" area of interest. Without such a "traverse science" application, potentially valuable science information could be missed, and onboard capability unused.

This kind of science application also fits quite easily into the above-described middleware infrastructure. First, the acquired images can be "published" to the onboard message store (thus making them immediately available to other applications), along with metadata such as acquisition time, location, etc. (as properties of the image object). An onboard science processing application can then "subscribe" to the sensor data, and returns results either as further metadata properties of the original image, or as a new processed object (e.g., a feature vector that also contains a reference to the original source object). The processed vectors can then be ordered by interest value. The scientist (who provided the feature processing algorithm) then subscribes to this ordered feature set, perhaps requesting only those above a certain level.

These results will be delivered semi-automatically (via some path created by the distribution middleware as connectivity and priority allow). This is quite a different approach than the current one, and should be considered only as a stimulant to conceive interesting new applications that might be feasible if such middleware were available and reliable.

Figure 2 shows a middleware-enabled environment providing capability to several example applications via the same few standardized middleware features (message transfer, local storage and query, prioritized delivery). More importantly, evolutions to this capability can easily be envisioned and simply implemented, perhaps even "on the fly". For example, the scientist could request (subscribe to) the original image for a particular feature

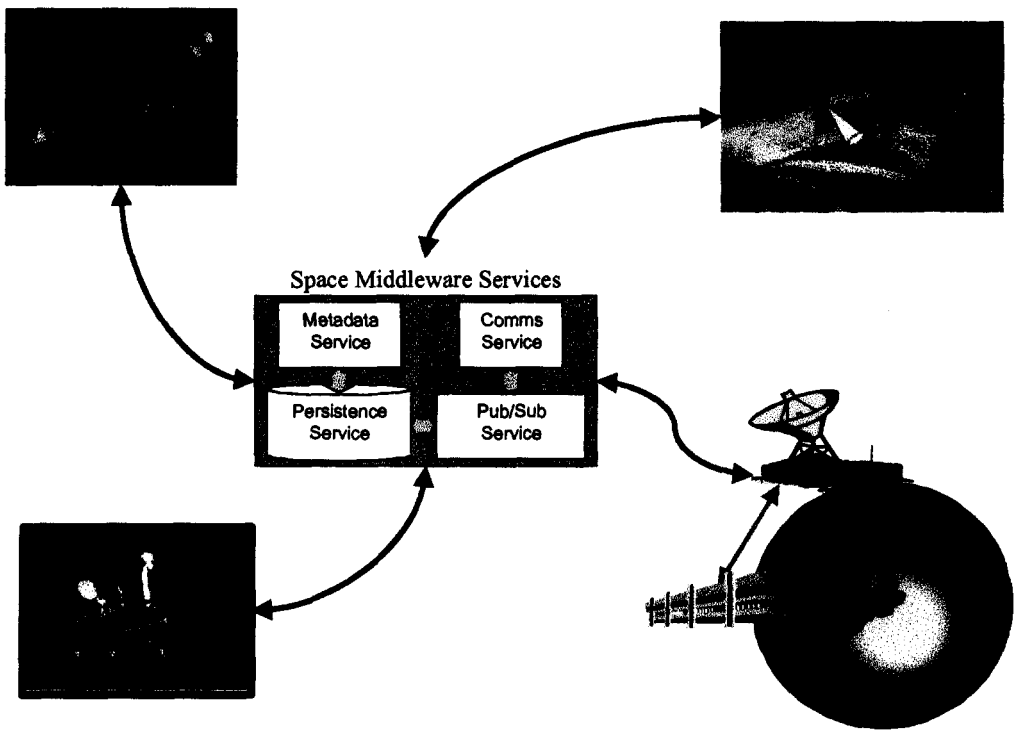

Figure 2: Space Middleware Environment

vector received, or could remotely modify the onboard processing algorithm to consider other factors, perhaps including information obtained from other sensors which may not even be on the same spacecraft. Temporary local storage can be semi-automatically released by deleting objects using such criteria as time to expire or lowest relative importance. Many such features can be implemented with little or no ground communication. This can preserve downlink bandwidth and increase quality of science delivered while simultaneously providing more rapid (perhaps even automated) response to science events. Finally, if sufficient regional storage provided (e.g., on an orbiting platform), information of value to other missions can be made available without ever making the round trip to Earth. For example, high-resolution mapping data obtained by an orbiting sensor could be utilized by a navigation algorithm on a rover. This is particularly interesting given that <1% of the mapping data produced by Mars Global Surveyor can be downlinked to Earth using current links.

4. DISCUSSION

Space-based middleware can help shift focus away from the details of point-to-point remote communication and towards a high-level service architecture with increased capability for automation and cooperation among space assets. A first step is to define high-level data objects and a mechanism to allow efficient exchange between producers and consumers interested in particular attributes of such objects. This also helps on-board applications to be insulated from inessential details (including the vagaries of the space communication). We believe these many of these details are best handled once (at the shared service layer), rather than every time (at the individual application level).

We distinguish the suggested space service architecture from the (required) underlying communication protocols.

Of course, such an architecture cannot be deployed in space without careful attention to its particular constraints (e.g., intermittent connectivity, limited resources such as bandwidth, power, etc.). Moreover, reliability and availability become paramount for services in the space domain, and need to be addressed much more carefully than is typical for internet-based services. For example, automated transparent service redundancy would be an important goal.

We also need to define interfaces to appropriate higher-level peer information services (as in the middle column of Figure 1), and to design these services to be layered appropriately despite the highly resource-constrained environment. Our suggested approach is to follow the incremental development of terrestrial web services, which are based on encapsulated components communicating via Internet protocols. As observed with the terrestrial Internet, functionality and benefits would increase as more services are deployed and more resources (spacecraft) co-operate.

There is significant ongoing work in areas such as IPN architecture, space communication standards, protocol development, onboard autonomy, distributed science. This description of space middleware is intended to be seen as working to assist this work, not to provide an alternative. It is perceived that the larger problem to be solved is not technological in nature, but cultural.

This work was performed at the Jet Propulsion Laboratory of the California Institute of Technology under NASA support. Original contributions of Anthony Barrett, Thom McVittie, and Brad Clement are acknowledged.

5. REFERENCES

1. V. Cerf et al., "Delay-Tolerant Network Architecture": March 2003: http://www.dtnrg.org/specs/draft_irtf_dtnrg_arch_02.pdf
2. K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets", IRB-TR-03-003, Feb., 2003: http://www.dtnrg.org/papers/IRB-TR-03-003.pdf
3. K. Scott, S. Burleigh, "Bundle Protocol Specification", Mar 2003, http://www.dtnrg.org/specs/draft_irtf_dtnrg_bundle_spec_00.pdf